Міністерство освіти і науки України
Національний університет «Одеська Політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Лісовський Михайло Андрійович,
студент групи РЗ-171

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Розробка мультисерверного корпоративного месенджера зі спеціальним захистом міжсерверного трафіку

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Кібербезпека

Керівник:
Стопакевич Олексій Аркадійович,
к.т.н., доцент

Одеса – 2022

Міністерство освіти і науки України
Державний університет «Одеська Політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти другий (магістерський)
Спеціальність 125 Кібербезпека
Спеціалізація, освітня програма Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ

_____
д.т.н.,проф. А.А.Кобозєва
____ _____202_р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Лісовському Михайлу Андрійовичу

1.Тема роботи: *Розробка мультисерверного корпоративного месенджера зі спеціальним захистом міжсерверного трафіку*,
керівник роботи *Стопакевич Олексій Аркадійович, к. т. н., доцент,*
затверджені наказом ректора від „____”_____ 20___ р. №_____ .
2.Зміст роботи: *аналіз сучасних корпоративних месенджерів, створення алгоритму роботи месенджеру та вибір мови програмування і середовища реалізації, розробка програмного забезпечення системи.*
3. Перелік ілюстративного матеріалу: *Типи з’єднання, Схематичне зображення процесу з’єднання серверів, Початкова сторінка одразу після запуску програми, Результат виконання програми, Вікно авторизації месенджера, Початковий стан месенджеру, Меню додавання контактів, Результат додавання контакту, Меню взаємодії з контактом, Листування з заблокованим користувачем, Налаштування профілю користувача, Форма реєстрації нового користувача.*

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Дата видачі завдання “_____”_____20___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз джерел з теми випускної кваліфікаційної роботи*	29.08.2022	*виконано*
2	*Обґрунтування вибору рішення. Збір даних*	15.09.2022	*виконано*
3	*Аналіз основних особливостей месенджерів*		*виконано*
4	*Аналіз принципів роботи мережевих з’єднанньь*	28.09.2022	*виконано*
5	*Розроблення мультисерверного корпоративного месенджера зі спеціальним захистом міжсерверного трафіку*		*виконано*
6	*Підготовка тексту роботи*	12.10.2022	*виконано*
7	*Підготовка презентації та доповіді*		*виконано*
8	*Попередній захист*	01.11.2022	*виконано*
9	*Нормоконтроль, рецензування*	12.11.2022	*виконано*
10	*Занесення роботи в електронний архів*	02.12.2022	*виконано*
11	*Допуск до захисту у завідувача кафедри*	19.12.2022	*виконано*

Здобувач вищої освіти _____ *Лісовський М.А.*

Керівник роботи _____ *Стопакевич О.А.*

АНОТАЦІЯ

Кваліфікаційна робота на тему “Розробка мультисерверного корпоративного месенджера зі спеціальним захистом міжсерверного трафіку” на здобуття другого (магістерського) рівня вищої освіти за спеціальністю 125 Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 13 рисунків, 1 додаток, 20 літературних джерел за переліком посилань. Робота виконана на 51 сторінці загального тексту і 41 сторінці основного тексту.

Метою роботи є створення з’єднання між окремими філіями корпорації без надання прямого доступу працівникам до мережі Інтернет.

У роботі проведено аналіз принципів У роботі проведено аналіз принципів створення з’єднання між серверами у мережі інтернет, що дозволило, на основі технології NAT bypass, визначити схему роботи месенджеру.

У результаті виконання кваліфікаційної роботи розроблено проект системи міжмережевого з’єднання серверів, що дозволило створити месенджер, який працює між кількома локальними мережами. Завдяки впровадженню системи вдалося створити канал передачі даних, який дозволяє передавати дані між окремими пристроями, розташованими у межах захищеного NAT.

Результати даної роботи можуть бути використані при розгортанні такої системи у корпоративній мережі.

МЕСЕНДЖЕР, З’ЄДНАННЯ, ЛОКАЛЬНА МЕРЕЖА, СЕРВЕР, ПЕРЕДАЧА ДАНИХ, ПОВІДОМЛЕННЯ.

ABSTRACT

Qualification work on "Development of a multi-server corporate messenger with special protection of inter-server traffic" for the second (master's) level of higher education in the specialty 125 Cybersecurity, specialization, educational program: Cybersecurity, contains 13 figures, 1 addition, 20 references. The work is completed on 51 pages of the general text and 41 pages of the main text.

The purpose of the work is to create a connection between separate branches of the corporation without providing employees with direct access to the Internet.

The paper analyzed the principles The paper analyzed the principles of creating a connection between servers on the Internet, which allowed, based on the NAT bypass technology, to determine the scheme of the messenger.

As a result of the qualification work, a project of a server interconnection system was developed, which made it possible to create a messenger that works between several local networks. Thanks to the implementation of the system, it was possible to create a data channel that allows data to be transferred between individual devices located within a protected NAT.

The results of this work can be used when deploying such a system in a corporate network.

MESSENGER, CONNECTION, LOCAL NETWORK, SERVER, DATA TRANSFER, MESSAGES.

ЗМІСТ

Вступ

Будь-які сучасні підприємства чи корпорації залежать від швидкості та надійності зв’язку між окремими підрозділами та офісами, що обумовлено необхідністю коригувати робочий процес у реальному часі та розподіляти роботу між різними підрозділами, для її пришвидшення.

У різних проміжках часу використовувались різні методи доставки інформації, спочатку це була фізична пошта, потім телеграф, після цього телефон, електронна пошта, файлообмінники та, врешті решт, месенджери.

Неможливо недооцінити внесок, який зробили месенджери, з самого моменту їх створення, і по цей час, месенджери значно прискорили та спростили процес обміну інформацією між людьми. Будь то звичайна бесіда між друзями, пересилання зображень котів, або ділова розмова, месенджери безумовно зробили цей процес значно легшим та більш очевидним, оскільки, на відміну від електронної пошти, користувач завжди знає коли його співбесідник був у мережі, чи прочитав він повідомлення та чи набирає він текст у відповідь.

Але, як і з усіма методами зв’язку, в будь-які часи, в них виникає вразливість, яка може поставити під великий ризик інформаційну безпеку організації, будь то виток чутливої інформації, її пошкодження або неавторизована зміна, що дуже негативно вплине як на репутацію, так і на фінансове положення організації. Звісно, кожна епоха мала свої ризики та методи боротьби з ними, наприклад - фізична пошта могла бути викрадена, знищена або підмінена на іншу, повідомленн через телеграф могли бути легко перехоплені, телефон підслуханий, електронна пошта підроблена.

Месенджери також не без гріха, оскільки у багатьох сучасних месенджерів існує або існувала купа вразливостей, від, здавалось би, незначних, як доступ до контактної книжки, так і дуже серйозних, як, наприклад, можливість для кіберзлочинця отримати повний контроль над пристроєм жертви.

Звісно, вразливості намагаються виправляти якомога швидше. Бо важливу пошту почали пересилати у захищених вагонах, повідомлення через телеграф

почали шифрувати, телефонний зв’язок почали пускати через спеціальні канали, електронна пошта почала перевіряти ідентичність відправника та фільтрувати фішинг. Месенджери також почали виправляти свої вразливості.

Але, як це часто і трапляється, месенджери загального призначення - програмний продукт, що постійно оновлюється і інколи можуть виникати нові вразливості. Як наслідок, використання таких месенджерів у корпоративних цілях може бути дуже ненадійним, вже не кажучи про те, що через такі месенджери існує ризик фішингових атак, які також можуть призвести до витоку інформації.

Звісно, більшість компаній й надалі використовують загальнодоступні сервіси, як, наприклад dropbox, для пересилання файлів, або використовують електронну пошту для розсилки повідомлень, що пов’язані з роботою, але це не відміняє усіх ризиків, які пов’язані з використанням такого ПЗ. Не кажучи вже про те, що прямий доступ до мережі Інтернет може пошкодити робочому процесу через те, що працівники можуть відволікатись на непов’язані з робочим процесом речі, будь то простий скроллинг стрічки новин, листування зі знайомими або перегляд зображень котів. Усі ці фактори дуже негативно впливають на загальну продуктивність підприємства чи організації, що може призвести до серйозних фінансових втрат або, потенційно, витоку чутливої інформації, що несе за собою ще більші фінансові та репутаційні втрати.

Саме ці фактори є основополагаючою причиною, через яку й було прийнято рішення створити месенджер корпоративного призначення, що дозволив би мати переваги LAN месенджерів та звичайних месенджерів, не втрачаючи ані в безпеці ані в швидкості передачі повідомлень, завдяки чому можливо було б створити ідеальну робочу середу, що буде дозволяти працівникам мати зв’язок з їхніми колегами, не надаючи їм прямого виходу до мережі Інтернет, що одразу вирішує питання фішингу, уважливості працівників та безпечності корпоративної таємниці та іншої чутливої інформації, що знаходиться в робочому обігу.

1. АНАЛІЗ СУЧАСНИХ КОРПОРАТИВНИХ МЕСЕНДЖЕРІВ

1.1 Аналіз LAN мереж та базових мережевих принципів

Щоб зрозуміти як працюють LAN месенджери, спочатку необхідно зрозуміти принципи, на яких вони засновані.

LAN (Local Area Network, локальна мережа) - це технологія зв’язку на обмеженій площі. Може працювати як в межах одного приміщення, так і в межах одного будинку. У порівнянні з WAN (Wide Area Network, глобальна мережа), LAN має більшу швидкість передачі даних. Але, як виходить з різниць у назвах, LAN має значно меншу площу покриття. Але, однією з головних переваг LAN є те, що мережа може бути як централізованою, так і децентралізованою, що надає більшу гнучкість при побудуванні такої мережі.

Основою будь-якої мережі передачі даних є мережеві протоколи. Основними протоколами передачі є TCP та UDP.

TCP (Transmission Control Protocol, протокол керування передаванням) - це протокол транспортного рівня, що керує передаванням даних у комп’ютерних мережах та який є основою моделі TCP/IP. Для надійності передачі даних від одного хоста до іншого, TCP встановлює логічне з’єднання між ними. Це означає що TCP відноситься до класу протоколів зі встановленням з’єднання.

Для того, щоб забезпечити надійне пересилання даних, протокол TCP розбиває потік даних на порції та додає до кожної з них заголовок з номером послідовності. Таким чином утворюються TCP-сегменти. Після чого, кожен сегмент інкапсулюється у IP-пакет та пересилається до отримувача. Він, в свою чергу, перераховуючи контрольну суму, перевіряє коректність отриманих даних у TCP-сегменті та переконується у тому, що попередні сегменти даних також були успішно отримані. Після чого отримувач надсилає запит до відправника про нове або повторне передавання сегменту, що, в свою чергу, є й підтвердженням того, що всі сегменти з номерами послідовності, меншими за номер запиту, були успішно надіслані.

UPD (User Datagram Protocol, протокол дейтаграм користувача) - ще один протокол транспортного рівня. На відміну від TCP, цей протокол працює без встановлення з’єднання. Цей протокол є одним з найпростіших та виконує обмін повідомленнями (дейтаграмами) без підтвердження та гарантії доставки. Але, попри цей та інші недоліки, протокол UDP є ефективним для серверів, що надсилають невеликі повідомлення великій кількості клієнтів, як приклад - протокол DHCP.

Для того, щоб розгорнути LAN месенджер, необхідно мати LAN мережу. Вони бувають у різних формах, дротові, бездротові, з централізованим хостом, децентралізовані. Але їхня загальна мета - об’єднання певного набору пристроїв в межах певної території у спільну мережу.

Централізований LAN - локальна мережа, побудована навколо одного серверу, що виконує усі основні обчислення. У такій мережі менш потужні робочі станції підключаються до серверу та надсилають свої запити до нього, замість того щоб виконувати їх самостійно.

Децентралізований LAN - локальна мережа, в якій навантаження розповсюджується між пристроями в мережі, замість навантаження центрального серверу. Такий підхід став набирати популярність разом зі збільшенням потужності комп’ютерів та ноутбуків до того рівня, коли продуктивність значно перевищує потреби більшості застосувань.

Ці два підходи до побудування локальної мережі мають певні відміни:

1. Масштабованість - здатність мережі до збільшення - у цьому пункті децентралізована мережа має перевагу, оскільки додавання нового пристрою до мережі збільшить як її розмір, так і потужність.
2. Надійність мережі - працездатність мережі у разі будь-яких технічних помилок - цей пункт, знов таки, надає перевагу децентралізованій мережі, оскільки навіть якщо один з пристроїв вийде з ладу, інші зможуть працювати й надалі, на відміну від централізованої мережі, де вихід серверу з ладу означає припинення роботи всієї мережі.

3. Приватність - наскільки важко відстежити трафік у мережі - також пункт, у якому перемагає децентралізована мережа, оскільки у ній трафік може переміщатися по різним маршрутам, що значно ускладнює відстеження трафіку.
4. Постійність - стабільність якості мережі - у цьому пункті перемагає централізована мережа, оскільки у ній адміністратор має обслуговувати та оновлювати лише один пристрій, що значно спрощує цей процес.
5. Ефективність - наскільки ефективно працює мережа - як би це не було дивно, але централізовані мережі більш ефективні, ніж децентралізовані, оскільки весь трафік проходить завжди однаково довгий маршрут.
6. Доступність - наскільки дешево побудувати мережу - і тут, знов ж таки централізована мережа перемагає, оскільки така мережа значно дешевша при встановленні.

Загалом, вибір типу локальної мережі залежить від бюджету та потреб, які має організація, оскільки комусь може вистачити класичної централізованої мережі, а для когось необхідно будувати децентралізовану мережу.

1.2 Аналіз роботи LAN месенджера

LAN месенджери використовують TCP для з’єднання, оскільки необхідно щоб повідомлення гарантовано надходили до серверу та від серверу до отримувача.

При встановленні підключення, TCP працює наступним чином:

1. Клієнт, який планує встановити з’єднання, посилає серверу сегмент з номером послідовності та флагом SYN (Synchronize sequence numbers, синхронізація номерів послідовності).
    a. Сервер отримує сегмент, запам’ятовує номер послідовності і намагається створити сокет для обслуговування нового клієнту.

   b. У разі успіху, сервер надсилає сегмент з номером послідовності і флагами SYN та ACK (Acknowledgment field is significant, задіяно “Номер підтвердження”) та переходить у стан SYN-RECEIVED.
   c. У разі невдачі, сервер надсилає сегмент з флагом RST (Reset the connection, обірвати з’єднання, очистити буфер)
2. Якщо клієнт отримує сегмент з флагом SYN, він запам’ятовує номер послідовності та надсилає сегмент з флагом ACK.
   a. Якщо клієнт водночас отримує й флаг ACK (природна поведінка), то він переходить до стану ESTABLISHED.
   b. Якщо клієнт отримує сегмент з флагом RST, він припиняє спроби з’єднатись.
   c. Якщо клієнт не отримує відповіді протягом 10 секунд, він знову повторює процес з’єднання.
3. Якщо сервер у стані SYN-RECEIVED отримує сегмент з флагом ACK, то він переходить у стан ESTABLISHED
   a. Інакше, після тайм-ауту, він зачиняє сокет та переходить у стан CLOSED

Цей процес називається “Триетапним рукостисканням” (Three way handshake), оскільки виконується усього у три сегменти, хоча процес з’єднання у чотири можливий, але на практиці достатньо трьох і це займає менше часу.

З’єднання у LAN месенджерах, та будь-яких інших програмах, реалізується за допомогою сокетів.

Сокет (Socket) - програмний інтерфейс, що забезпечує обмін даними між процесами, які можуть виконуватись як на одній ЕОМ, так і на різних ЕОМ, що пов’язані між собою мережею. Сокет є абстрактним об’єктом, що представляє собою точку з’єднання.

Кожен процес може створити “слухаючий” (серверний) сокет і прив’язати його до якогось порту операційної системи. Процес, що “слухає”, зазвичай, знаходиться у циклі очікування і “прокидається”, коли з’являється нове з’єднання.

Фактично, сокет є інструментом керування портом для процесу, який дозволяє процесу “слухати” або передавати будь-що через вказаний порт.

Порт - мережевий інтерфейс, призначений для створення з’єднання між процесами з інших ЕОМ. Одночасно на одному пристрої може бути виділено до 65536 портів (нумерація від 0 до 65535).

В залежності від типу LAN месенджеру, можливі різні варіанти роботи. Наприклад - це може бути LAN чат, у який пише велика кількість людей одночасно, це може бути чат 1 на 1 або набір чатів з різними користувачами.

Також, LAN месенджери можуть працювати без центрального серверу, завдяки P2P (Peer-to-peer) з’єднанню. (рис 1.1)

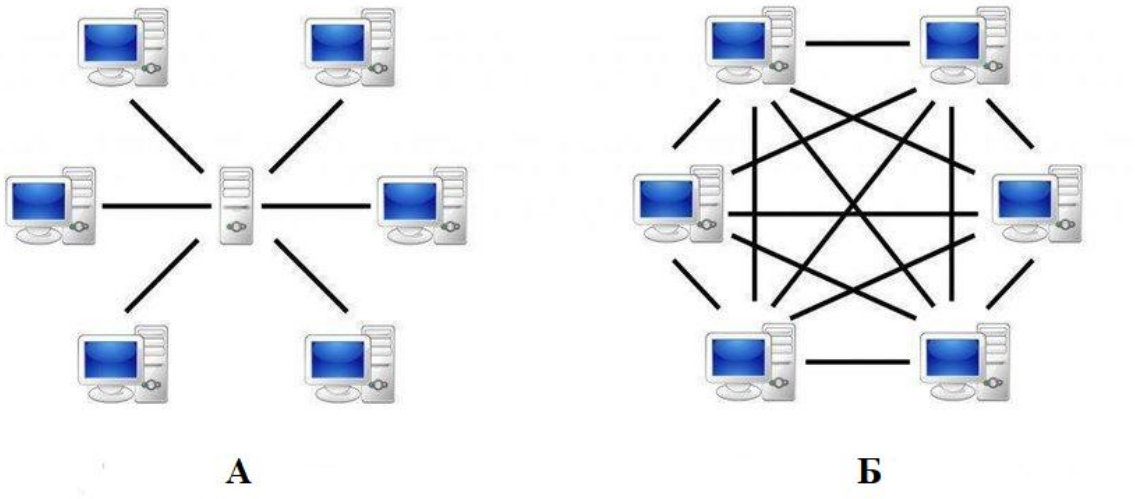


Рисунок 1.1 Типи з’єднання у локальній мережі, де: а) LAN месенджер з сервером; б) LAN месенджер з використанням P2P

Різниця між зв’язком через сервер та P2P у тому, що для мережі P2P кожен пристрій має з’єднуватись з усіма іншими окремо.

LAN месенджери та чати бувають різного виду, починаючи від найпростіших, де в якості ідентифікатору користувача використовується його IP адреса, до більш просунутих, де в користувача є юзернейм та можливість писати до окремих користувачів, що знаходяться в межах LAN мережі.

Відповідно, у випадку централізованого LAN месенджеру, усі повідомлення надсилаються до серверу, який в свою чергу пересилає повідомлення одержувачу. У випадку децентралізованого LAN месенджеру, кожен клієнт самостійно надсилає повідомлення до одержувача, пересилаючи повідомлення до того порту, на якому було встановлено зв’язок.

У випадку LAN месенджерів, найбільш поширеною і найбільш популярною є саме децентралізована, оскільки дуже просто, маючи декілька комп’ютерів та/або ноутбуків, створити таку мережу без необхідності придбати окремий пристрій, що буде виступати в якості серверу. До того ж, у такій конфігурації зв’язок між іншими абонентами не буде втрачено, навіть якщо один з них вийде з ладу, оскільки у такій мережі прямої залежності від одного пристрою немає, що значно підвищує її надійність.

1.3 Аналіз існуючих корпоративних месенджерів

У сьогоденні для корпоративного використання застосовують різні месенджери та чати. До цього списку входять як і відомі всім месенджери, такі як Telegram, WhatsApp та Viber, так і менш відомі серед звичайних користувачів, як, наприклад, Slack або Microsoft Teams.

Важливим фактором є те, що більшість використовуємих месенджерів працюють або виключно в межах локальної мережі або виключно з використанням інтернет-підключення. І, хоча, месенджери, що працюють з інтернет-підключенням, мають позитивні сторони, однією з проблем є саме надання доступу до мережі Інтернет зі сторони корпоративної мережі, що може поставити під великий ризик інформаційну безпеку підприємства.

Дуже важливо розрізняти звичайні месенджери від корпоративних. Так, звісно, у корпоративному просторі можуть застосовуватися обидва варіанти, але корпоративні месенджери, в першу чергу, спрямовані саме на використання для корпоративної та командної праці, що значно спрощує процес комунікації під час виконання робочих завдань.

Розглянемо деякі з існуючих месенджерів, що використовуються в сучасних корпораціях.

Одним з варіантів виступають такі месенджери як: Viber, Telegram, Facebook Messenger та WhatsApp. Усі вони поєднані тим, що їх першочергове призначення - спілкування через мережу Інтернет. Але кожен з цих месенджерів між собою відрізняється. Наприклад, Telegram дозволяє дуже високу ступінь кастомізації інтерфейсу, Facebook Messenger дозволяє легко спілкуватись з людьми, які входять до списку друзів у соціальній мережі Facebook, тощо.

Месенджер Telegram був створений Павлом Дуровим, колишнім засновником соціальної мережі VK, що потім була насильно в нього вилучена. Під час створення Telegram, Павло вирішив створити надійний та безпечний месенджер, який дозволить безпечно спілкуватись та пересилати зображення, в тому числі котів, а також інші медіафайли, такі як музику, відео, тощо.

Telegram дозволяє як чати формату 1 на 1, так і групові чати, а також - групи, у яких адміністратор може публікувати необхідні новини, але крім користувачів з привілеями, ніхто більше не може публікувати туди будь-що. Такий тип чату використовується для публікації новин або будь-якої іншої інформації.

Інтерфейс Telegram - один з найгнучкіших інтерфейсів, серед доступних користувачам месенджерів, оскільки він дозволяє змінити кольорову схему, фонове зображення чатів, сортувати чати по папкам, та інше.

Загальних відмін в інших месенджерів небагато, але серед них є наступні: Viber, WhatsApp та Facebook Messenger не дозволяють сортувати канали по окремим категоріям та детально налаштовувати кольорову схему. WhatsApp та Facebook Messenger також не дозволяють створювати канали для новин, а в самого Facebook Messenger взагалі відсутня можливість внесення візуальних змін до чату, окрім вибору між світлою і темною темами.

Але, оскільки всі ці месенджери є месенджерами загального призначення, це означає що працівники можуть використовувати їх не тільки для роботи, вони можуть писати своїм друзям, дивитись меми або зображення котів, що, хоч і дуже

приємно, є відволікаючим фактором та може негативно відобразитись на продуктивності.

Ще один варіант - використання месенджерів, націлених саме на корпоративні потреби. До таких входять Slack, MyChat та Microsoft Teams.

Slack - месенджер, призначений для корпоративного використання, створений компанією Slack Technologies.

У цьому месенджері користувачі можуть комунікувати за допомогою відео та аудио дзвінків, а також за допомогою текстових повідомлень, медіа та файлів, використовуючи персональні чати або спільноти, відомі як “workspace” (робочий простір).

Особливості Slack:

1. Teams (команди) - дозволяє командам, спільнотам або групам приєднатися до “workspace” через URL-посилання, яке було створено адміністратором або володарем команди.
2. Обмін повідомленнями
   a. Публічні канали. Доступні всім користувачам у “workspace”; можуть бути конвертовані у приватні чати.
   b. Приватні чати. Використовуються для приватного обміну повідомленнями в малих підгрупах. Не можуть бути конвертовані у публічні чати.
   c. Прямі повідомлення. Дозволяють користувачам відправляти повідомлення конкретному користувачу, замість групи людей. Можуть включати до дев’яти користувачів одночасно. Після створення може бути конвертовано у приватний чат.
3. Huddles (тусовки). Huddle це спеціальний чат, у якому користувачі спілкуються за допомогою голосового зв’язку, а також можуть демонструвати екран та малювати на ньому під час демонстрації. Один Huddle може мати до 2 (50 у преміум версії) користувачів одночасно.
4. Інтеграції. Slack має інтеграції з GitHub, Dropbox, Google Drive, Google Calendar, тощо. Також Slack підтримує інтеграції, створені спільнотою.

5. API. Slack надає користувачам доступ до application programming interface (інтерфейс програмування додатка) для користувачів, що дозволяє створювати додатки та автоматизовувати процеси.
6. Slackbot. Slack дозволяє користувачам додавати та налаштовувати чат-ботів, які можуть надсилати нагадування або повідомлення, а також надавати спеціальні відповіді на різні ключові фрази, тощо.

Нажаль, зі Slack траплялись й інцеденти. Один з них трапився у Травні 2015 року. Slack повідомив що вони були зламані протягом чотирьох днів у Лютому того ж року. Згідно зі звітом команди, зловмисники викрали деякі дані, пов’язані з користувачами, серед цих даних були електронні пошти, юзернейми, захешовані паролі, номери телефонів та Skype ID. У відповідь на атаки, Slack додали двофакторну аутентифікацію до своїх сервісів. Також, у 2021 та 2022 роках, Slack зазнав значних збоїв в роботі, через що користувачі не мали змоги скористатися будь-якими функціями.

Microsoft Teams - прямий конкурент Slack. У Microsoft Teams є інтеграція користувачів, вмісту та засобів, що необхідні команді для ефективної роботи. Усе це об’єднується в спільному робочому середовищі, до якого також входять: чат для нарад, файлообмінник та корпоративні програми.

Особливості Microsoft Teams:

1. Чати. MS Teams дозволяє користувачам з’єднуватись за допомогою двостороннього постійного чату з одним або кількома учасниками. Учасники можуть відправляти текстові повідомлення, емоджі, стікери та гіфки, а також посилання та файли.
2. Команди. MS Teams дозволяє робити внесок до робочого простору спільнотам, групам або командам. У цьому робочому просторі усі повідомлення та цифровий контент по конкретному топіку поширюється між усіма користувачами. Члени команд можуть приєднатися за допомогою посилання, що було створено адміністратором команди або іншим користувачем.

3. Канали. Канали дозволяють комунікацію без необхідності надсилати SMS повідомлення або використовувати електронну пошту. Користувачі можуть відповідати до постів за допомогою тексту, зображень, гіф-зображень або мемів. Приватні повідомлення надходять до конкретної особи, а не до всього чату.
4. Заміна телефону. Особливість, доступна в одному з дорожчих рівнів ліцензії, що дозволяє підключення до телефонної мережі, завдяки чому можна використовувати MS Teams як телефон, робити та приймати дзвінки, в тому числі проводити конференційні дзвінки.
5. Meeting. Форма зв’язку, при якій користувачі можуть використовувати аудіо та відеозв’язок, а також текстовий чат. Такі дзвінки також можуть бути заплановані заздалегідь.
6. Інтеграції. MS Teams має інтеграції через інтеграційну платформу Microsoft AppSource.

Microsoft Teams також не без вразливостей. Згідно зі статтею The Daily Swig, Microsoft Teams має деякі дуже серйозні вразливості, з яких, на цю мить, була виправлена лише одна - вразливість, завдяки якій у версії програми на ОС Android міг статися витік IP-адреси користувача. А до актуальних вразливостей входять можливість підміни URL-посилань, що може трапитись на WEB та десктопних версіях, а також вразливість до DoS атак на пристроях під керуванням ОС Android.

Останній, але не останній за значенням, месенджер - MyChat, розроблений в Україні командою Network Software Solutions. Цей месенджер розповсюджується за ліцензією Shareware. Пробна версія підтримує до 20 одночасних з’єднань з сервером, а також демонструє рекламні банери.

MyChat побудований за технологією клієнт-сервер, що дозволяє цьому месенджеру працювати як у мережі Інтернет, так і у локальній мережі. Загалом, це стандартний LAN-чат, який має простий користувацький інтерфейс, можливість додавати зображення до тексту, пересилати файли, тощо.

В цього чату також є недоліки, серед них - сервер месенджеру може працювати лише під управлінням ОС Windows не раніше версії Windows 2000, що значно ускладнює інтеграцію такого месенджеру у деякі системи. Також, сервер не має користувацького інтерфейсу, що також робить його більш складним та менш комфортним у використанні.

Висновок: у ході аналізу роботи LAN месенджерів, LAN мереж та існуючих корпоративних месенджерів, було встановлено що існуючі месенджери або працюють виключно через мережу Інтернет, що може становити загрозу інформаційної безпеки організації, або виключно у локальній мережі, що значно обмежує можливості застосування таких месенджерів.

Виходячи з цього, мною було встановлено мету розробити корпоративний месенджер, який буде поєднувати безпечність LAN месенджерів разом з можливістю з’єднувати віддалені департаменти між собою через мережу Інтернет, забезпечуючи надійний та безпечний зв’язок між ними.

2. СТВОРЕННЯ АЛГОРИТМУ РОБОТИ МЕСЕНДЖЕРУ ТА ВИБІР МОВИ ПРОГРАМУВАННЯ І СЕРЕДОВИЩА РЕАЛІЗАЦІЇ

2.1 Створення алгоритму роботи месенджеру

Алгоритм роботи месенджера складається з двох частин, кожна з яких поділяється на декілька основних компонент і кожна з цих компонент є необхідною для роботи месенджеру у повному обсязі.

Перша частина це сервер, який пов’язує локальну мережу підприємства з його віддаленими локальними мережами. Перша компонента серверу відповідає за з’єднання між сервером та клієнтом. Фактично, це просто логічний зв’язок, побудований за допомогою сокетів, згідно з принципами роботи TCP.

Друга компонента відповідає за з’єднання серверу з іншими серверами. Ця компонента повинна не тільки створювати прямий зв’язок з іншим сервером, вона також має відстежувати, чи є інша сторона дійсно сервером, з яким вона має створити з’єднання. Для цього можуть використовуватись сертифікати, які будуть перевірятись при встановленні підключення. Схематично процес підключення продемонстровано на рисунку 2.1.

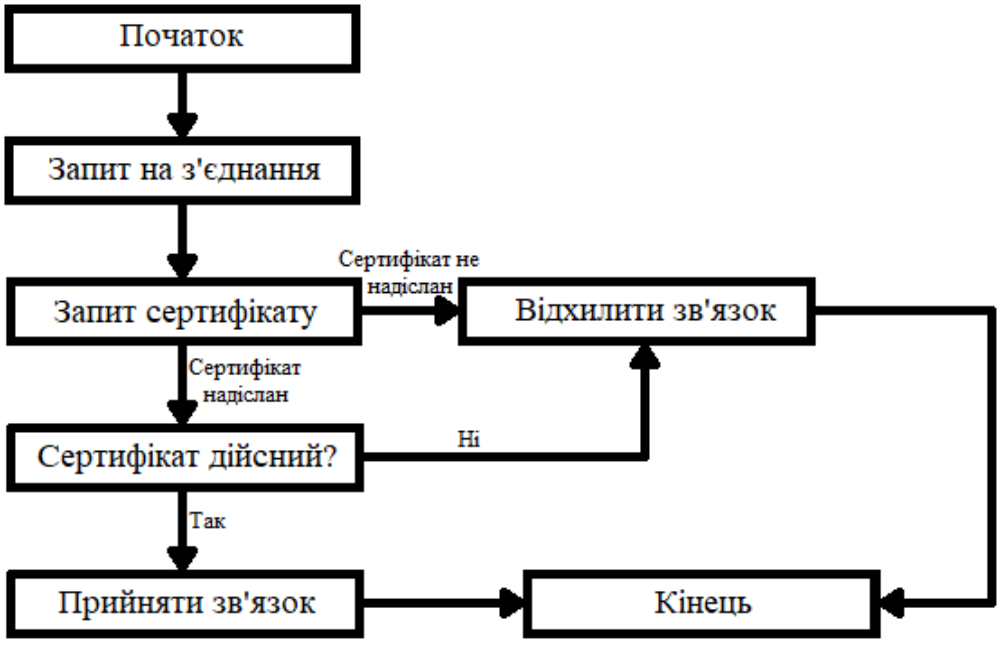


Рисунок 2.1 Схематичне зображення процесу з’єднання серверів.

Третя компонента серверу - це алгоритм, який обробляє отримане повідомлення від клієнта або від іншого сервера та приймає рішення куди воно має бути надіслано далі. Якщо це повідомлення від користувача і воно направлено до користувача у іншій мережі, сервер перенаправить це повідомлення до іншого серверу. Якщо це повідомлення від користувача до іншого користувача у цій самій мережі, сервер перенаправить повідомлення до іншого користувача. Якщо це повідомлення від сервера з іншої мережі до користувача у цій мережі, сервер перенаправить повідомлення до користувача.

Четверта компонента - це база даних користувачів, де зберігаються дані про кожного з користувачів - логін; ім’я; хешований пароль; мережа у якій знаходиться цей користувач; його мережевий статус. Завдяки цій базі, користувач може зайти у свій акаунт з будь-якої з локальних мереж корпорації, що спрощує процес переміщення працівників з відділу до відділу.

П’ята компонента серверу дозволяє серверам обмінюватися даними щодо нових користувачів та дані про них, такі як мережевий статус, який може бути:

1) в мережі;
2) був(-ла) в мережі n хвилин/годин назад;
3) був(-ла) в мережі о хх:хх учора;
4) був(-ла) в мережі n днів тому;
5) набирає повідомлення.

При тому, додаванням нових користувачів займається мережевий адміністратор, що дозволяє уникнути проблем, пов’язаних з логіном, який може виявитись зайнятим іншим працівником.

Друга частина це клієнтська сторона месенджеру. Перша її компонента відповідає за з’єднання клієнта з сервером, тому працює вона так само, як і з боку серверу.

Друга компонента клієнтської сторони відповідає за контроль списку контактів. Клієнт має змогу додавати до контактів інших працівників організації, з якими він працює у одній команді або над спільним проектом. Фактично це база

даних, яка зберігає логін; ім’я користувача; відділ до якого він підключений; та його мережевий статус. Ці дані вона отримує від серверу.

Третя компонента відповідає за шифрування та передачу текстових повідомлень від клієнта до сервера. При передаванні повідомлення, клієнт надсилає серверу повідомлення у якому вказує одержувача, відправника та шифроване повідомлення. Повідомлення шифрується для додаткової безпеки, на випадок, якщо внутрішня частина мережі буде скомпроментована.

Четверта компонента це FTP сервер, який відповідає за відправку та отримку файлів від клієнта до сервера. При цьому, файл, що надсилається до серверу, також шифрується з тих самих міркувань, що і у третій компоненті.

П’ята компонента займається постійним оновленням інтерфейсу користувача для відображення нових повідомлень, чатів, мережевих статусів контактів, тощо.

Загальна схема роботи месенджеру зображена на рисунку 2.2.

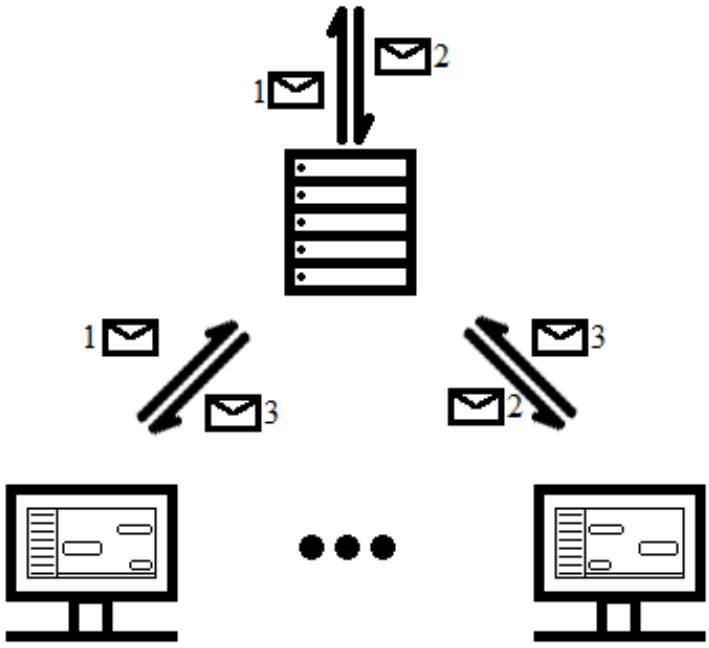


Рисунок 2.2 Загальна схема роботи месенджеру, де 1 це повідомлення від користувача до іншої мережі, 2 від іншої мережі до користувача, 3 між користувачами в межах мережі.

2.2 Вибір мови програмування

Зараз існує велика кількість різних мов програмування, які вирішують різні задачі. Мова програмування може бути сфокусована на інтерфейсі, на виконанні команд, або на двох аспектах одразу.

Важливо розуміти для чого найкраще використовувати ту чи іншу мову програмування, щоб якомога краще оптимізувати процес написання програмного продукту.

Виходячи з цього, мною було розглянуто декілька різних мов програмування, серед яких присутні python, javascript та c/c++. Також мною було розглянуто їхні можливості, пов’язані зі створенням графічних інтерфейсів та проведено порівняння цих можливостей з WEB-інтерфейсом.

Почнемо з найпопулярнішої нині мови програмування - python. Мова програмування python є інтерпретованою об’єктно-орієнтованою мовою високого рівня. Python підтримує декілька парадигм, серед них:

1) об’єктно-орієнтоване програмування
2) процедурне програмування
3) функціональне програмування
4) аспектно-орієнтоване програмування

Також, Python підтримує модулі та пакети модулів, що сприяє модульності програм та можливості повторного використання коду. Інтерпретатор python та усі стандартні бібліотеки доступні на всіх основних платформах (Windows, MacOS, Linux).

Стандартний набір бібліотек для python дозволяє користувачам взаємодіяти з мережевими інтерфейсами, керувати файловою системою пристрою, використовувати різноманітні мережеві протоколи, такі як HTTP або FTP, також є модулі для роботи з XML, модулі криптографії, тощо. Стандартна бібліотека python є однією з найбагатших бібліотек, серед сучасних мов програмування.

Також, python підтримує додаткові модулі, розроблені іншими користувачами. Функціонал таких модулів дуже широкий, в них входять як модулі типу numpy, scipy, які часто використовуються для різноманітних

обчислень, так і модулі які дозволяють працювати з майже будь-якими типами баз даних, модулі що надають повний доступ до win32 api, також є модулі, що дозволяють взаємодіяти з іншими мовами програмування, використовувати більш просунуті мережеві протоколами, різноманітні криптографічні алгоритми, створювати інтерфейси різного рівня, у тому числі за допомогою поєднання python та html/css.

Загалом, python дозволяє дуже високий рівень кастомізації усіх елементів програми, що дозволяє дуже широку варіативність у якості програмних продуктів.

Також, python має цілу низку модулів, що дозволяють створювати на його базі різноманітні нейронні мережі, які можуть виконувати як базові функції, так і задачі на кшталт синхронізації рушень губ та аудіо, що дозволяє, наприклад, коригувати те, що каже репортер, у випадку будь-яких помилок під час запису.

Усе це робить python майже очевидним вибіром, але, заради справедливості, необхідно розглянути й інші перелічені мови.

Далі ми розглянемо c/c++ - статично типізовані мови програмування, що компілюються. На цьому визначенні починається головна відмінність цих мов програмування від python. Мова, що компілюється, це мова, яка переводиться у машинний код перед тим, як вона виконується. На відміну від цього, python є мовою що інтерпретується, що означає що код програми виконується поступово спеціальним інтерпретатором. Це означає що для запуску програми на c/c++ потребується більше часу, але виконання програми на python протікає повільніше.

Статична типізація означає що у мові програмування тип змінної визначається одразу, та не може бути змінений після цього. Це також відрізняється від python, де кожна змінна за час виконання програми може кілька разів змінити свій тип, тобто змінна може спочатку бути списком, потім стрічкою, а потім числом, що дозволяє більш гнучкий підхід до створення програми.

Мови програмування c/c++ мають своє місце у багатьох вбудованих системах та суперкомп’ютерах, та є основною мовою програмування у таких напрямках, як automotive та IoT.

Мова С орієнтована на процедурне програмування і в ній дуже важко використовувати об’єктно-орієнтований підхід, що зменшує кількість можливих сценаріїв використання, але все ще дозволяє цій мові бути достатньо корисною у її сфері застосування.

Одна з найголовніших особливостей мови С це вказівники, завдяки яким можливо отримати доступ до пам’яті пристрою, вказувати на змінні, створювати посилання на значення змінних для використання у функціях (загалом до функцій значення можливо передавати напряму, без необхідності робити зайві дії, що також є дуже корисною особливістю).

Технічно, за допомогою дуже громіздкої методіки, яка базується на фактичній поліморфності вказівників мови С, можливо реалізувати принципи ООП у цій мові програмування. Але, наприклад, реалізація функціонального програмування у мові С неможлива за жодних обставин.

Також, мова С має різноманітні бібліотеки, які додають різну функціональність, яка не присутня у цій мові за замовчуванням. До такої функціональності входять автоматичні прибиральники сміття (функції, що очищають пам’ять від елементів, що більше не існують, що дозволяє звільнити додаткові об’єми пам’яті для подальшого використання), підтримка багатозадачності, мережеві функції, тощо.

Мова програмування С є основою для деяких інших мов програмування, серед яких є C++, C# та Java. До того ж, мова програмування С має дуже великий вплив на розвинення індустрії програмного забезпечення в цілому.

Деякий час С дуже активно розвивалася, але з часом це пішло на спад і останні суттєві зміни відбулися у версії C11, до якої додали реалізацію потоків та атамарні типи. Найновітніша існуюча версія, C18, несе у собі лише виправлення помилок версії C11.

У той самий час, мова програмування C++ водночас підтримує мови такі парадигми програмування, як процедурне, об’єктно-орієнтоване та узагальнене програмування. На відміну від свого попередника, мови С, у C++ багато уваги надано саме об’єктно-орієнтованому та узагальненому програмуванню.

Узагальнене програмування - це форма програмування, яка полягає у такому описі даних та алгоритмів, яке можливо застосовувати до різних типів даних, не змінюючи при цьому сам опис. Фактично, це поділ структур даних та алгоритмів, виконаний через використання абстрактного опису вимог до них.

Синтекс C++ успадкований від C, оскільки при розробці C++ передбачалась зворотна сумісність з C. Тим не менш, C++ не є надмножиною мови C, оскільки, не дивлячись на те, скільки програм, написаних мовою C, можуть бути трансльовані як компіляторами C, так і компіляторами C++, це не включає усі можливі програми, що були написані мовою C.

Загалом, C++ є самостійною мовою, яка має часткову сумісність з C, на відміну від Objective C, що є об'єктно-орієнтованою версією C і, як наслідок, її надмножиною.

З нововведень, мова C++, починаючи з версії 11, має вбудований regex, функціональну компоненту, що дозволяє виконувати пошук по тексту за заданими параметрами, тип bool, можливість перетворювати типи у вигляді функцій, вбудовану підтримку посилань (що доводилось емулювати за допомогою вказівників у мові C), також з'явилась можливість додавати коментарі у вигляді //, зі збереженням коментарів типу /* */, які були у мові C, було додано й аргументи за замовчуванням, які можуть використовуватись, якщо під час виклику функції не було надано аргументів.

Усе це робить мову C++ дуже якісною еволюцією мови C та розширює її потенціал у використанні для різних сценаріїв, одним з основних, як і у C, є використання в якості мови програмування для взаємодії з апаратним забезпеченням різного типу, від мікрочіпів у мікрохвильовій пічці, до обладнання космічних кораблів.

Але, навіть так, обидві ці мови не часто використовуються у backend.

Далі йде мова JavaScript, яка є обов'язковою частиною будь-якого сучасного сайту або веб-додатку.

JavaScript підтримує декілька парадигмів програмування, серед яких є об’єктно орієнтована, імперативна та функціональна. Імперативна парадигма передбачає наступне:

1. У вихідному коді записуються інструкції (команди);
2. Ці інструкції повинні виконуватись поступово.

Мову JavaScript використовують як міст між об’єктами веб-сторінки та backend, у якості якого вона показує дуже гарні результати.

JavaScript дозволяє динамічно оновлювати веб-сторінку, прикладом чого є динамічна модифікація об’єктів сторінки у відповідь на зміну розміру вікна та динамічне додавання та видалення об’єктів на сторінці, створення каруселей зображень, тощо.

Також, завдяки JavaScript, можливо інтегрувати елементи з інших веб-сторінок, наприклад інтеграція відеозаписів з YouTube або інтерактивної мапи з Google Maps, та інші потенційні взаємодії з API різних сервісів, на кшталт сайтів з погодою, новинами, тощо.

Однак, JavaScript не має вбудованого API, який міг би працювати з файловою системою та управлінням потоками вводу-виводу. Також відсутній й стандартні інтерфейси до веб-серверів та баз даних. Усе це робить JavaScript менш комфортним при роботі у backend, що є суттєвим недоліком.

JavaScript може знаходитись безпосередньо у файлі HTML, у окремих елементах веб-сторінки, або у окремому файлі, який підгружається до сторінки під час її завантаження. Також, JavaScript відповідає за додаткові діалогові та спливаючі вікна, що дозволяє покращити взаємодію користувача з веб-сторінкою або веб-додатком.

Загалом, JavaScript позиціонується як мова користувацької сторони, оскільки основним застосуванням цієї мови є саме виконання клієнтської частини скриптів, які дозволяють веб-інтерфейсу спілкуватись з backend частиною додатку або віддаленого серверу.

Окремо слід відзначити особливості у підході до створення користувацького інтерфейсу у перелічених раніше мовах програмування, а саме python, C та C++.

У мові програмування python існує велика кількість різних модулів для створення інтерфейсу, серед них є такі як tkinter, PyQT, та модулі, які дозволяють пов'язати код python із веб-інтерфейсом, завдяки чому можливо створити веб-додаток.

Модуль tkinter є одним з модулів, що входять до стандартної бібліотеки та дозволяє створити інтерфейс, в якого буде крос-платформна підтримка, що дозволяє легко створювати додатки з простими інтерфейсами.

Але, звісно, стандартного інтерфейсу tkinter частіш за все не вистачає для виконання певних завдань, тоді використовуються бібліотеки типу PyQT та її додаток PySide, які дозволяють створювати більш складні інтерфейси та додавати до них різноманітні елементи, як, наприклад, графіки, завдяки яким можливо відображати важливу інформацію у легкому для розуміння вигляді.

Інколи, однак, не вистачає навіть таких модулів як PyQT, тоді використовується зв'язка python з веб-інтерфейсом, яка виконується за допомогою JavaScript та спеціальних бібліотек, які дозволяють python звертатись до JavaScript та обробляти звернення від нього, завдяки чому стає можливим створення адаптивних та комфортних користувацьких інтерфейсів.

Для мови C існує модуль GTK, який дозволяє створювати користувацькі інтерфейси. Саме завдяки мові C був створений інтерфейс ОС Windows, тому для цієї ОС можливо створити додаток з інтерфейсом, написаний повністю цією мовою.

Мова C++ має власну бібліотеку, націлену на створення графічних інтерфейсів - Visual C++, завдяки якій створена переважна більшість сучасних графічних інтерфейсів та програм, у тому числі відео-ігр, фото та відео редакторів, тощо.

Слід зазначити також, що до будь-якої з цих мов можливо підключити інтерфейс, який знаходиться у браузері та використовувати ці мови у якості backend, але для деяких з цих мов це може бути дуже серйозним викликом.

Враховуючи все вищезазначене, мною було обрано зв'язку python з інтерфейсом реалізованим завдяки html/css та зв'язаним з python через javascript.

2.3 Вибір середовища реалізації

Оскільки для розроблення програми використовується більше однієї мови програмування, необхідно вирішити питання середовища реалізації для кожної з них. Це важливо, оскільки кожна з мов має свої особливості тому, з метою зниження завантаженості, необхідно обрати окреме середовище для python та окреме середовище для html/css/javascript.

Почнемо з огляду існуючих інтерпретаторів python, оскільки переважна частина усієї програми буде реалізована саме у цій мові.

Однією з IDE (Integrated Development Environment) для python є IDLE (Integrated Development and Learning Environment), яка поставляється разом з python. Ця IDE є дуже простою з точки зору оформлення та не потребує будь-яких додаткових завантажень. Вона також є крос-платформовою, оскільки вона працює на всіх операційних системах, які підтримують python.

Також, дуже поширеною є PyCharm. Ця IDE поширюється за моделлю freemium, яка надає доступ до переважної кількості функціоналу IDE, але за деякі додаткові функції необхідно заплатити, щоб отримати до них доступ. PyCharm також працює на всіх основних ОС, що робить цю IDE дуже доступною.

Для python також можна використовувати Visual Studio Code, яка поширюється безкоштовно і працює як на ОС Windows, так і на MacOS. VS Code також має підтримку JS та CSS/HTML і дозволяє виконувати код в середині, що значно спрощує процес створення програмного продукту.

Серед інтерпретаторів python також існують інтерпретатори на кшталт PyScripter, які дозволяють писати та виконувати код без використання

додаткового програмного забезпечення. Але PyScripter доступен лише у ОС Windows, що хоч і є недоліком, не є надто великою проблемою.

Саме тому для python мною було обрано PyScripter, оскільки, на відміну від VS Code, він є менш навантаженим як з точки зору інтерфейсу, так і додаткових елементів, які також навантажують систему при використання VS Code. А на відміну від PyCharm, PyScripter може самостійно інтерпретовувати написаний код, тим самим спрощуючи процес відладки роботи програми.

Для javascript/css/html існує безліч різних текстових редакторів, починаючи від стандартних для операційних систем, таких як блокнот у ОС Windows, закінчуючи IDE на кшталт Visual Studio.

У цьому випадку, головну різницю становить можливість спостерігати зміни у реальному часі, де такі IDE як Sublime Text або VS мають перевагу, оскільки вони дозволяють створити окремий процес вікна браузеру, у якому в режимі реального часу буде оновлюватись сторінка. З іншими IDE доведеться самостійно запускати створену html сторінку у браузері і самостійно оновлювати сторінку браузеру для того, щоб побачити зміни.

Враховуючи усе, мною був обраний Sublime Text, який розповсюджується за моделлю freemium і який дозволяє відстежувати зміни у реальному часі, але, на відміну від VS, має відносно малу вагу та не навантажує систему.

Висновок: мною було розроблено алгоритм роботи корпоративного месенджеру, проаналізовано мови програмування та обрано зв’язку python + html/css/js, як саму адаптивну та гнучку, а також було обрано середи програмування для python та для html/css/js, у яких буде розроблено backend та frontend месенджеру.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Розробка алгоритму роботи програми

Для реалізації роботи месенджеру необхідними є наступні бібліотеки мови python:

1. Socket
2. Eel
3. libzt

Модуль socket є стандартним модулем мови програмування python. Цей модуль дозволяє програмі використовувати сокети для з’єднання з іншими програмами, які можуть знаходитись як на тому ж самому пристрої, так і на віддаленому.

Цей модуль дозволяє додатку слухати події на певному порті або використовувати випадковий порт для виконання підключення до іншого додатку, використовуючи протоколи IPv4 та IPv6, а також створювати з’єднання через Bluetooth.

Завдяки модулю socket, створення клієнт-серверних додатків, здатних працювати як в межах локальної, так і глобальної мереж, стає набагато легшим, що дозволяє сфокусуватись на вирішенні інших питань, пов’язаних з розробкою додатків такого типу.

Модуль Eel дозволяє створити взаємодію між python та javascript, завдяки чому можливо створити працездатний веб-інтерфейс, що значно підвищує можливості з точки зору адаптивності та можливості модифікувати зовнішній вигляд інтерфейсу, не створюючи занадто заплутаний програмний код.

Даний модуль не входить до стандартної бібліотеки python, тому для його використання необхідно його завантажити за допомогою `pip install eel`, після чого його буде можливо використовувати при написанні коду програми та під час відлагодження написаного коду.

Третій вказаний модуль - libzt - є бібліотекою, що дозволяє створювати з’єднання за допомогою протоколу zerotier. Для того, щоб ця бібліотека могла функціонувати, необхідно скористатись командою `pip install libzt`.

Zerotier це протокол, що дозволяє створювати захищене peer-to-peer з’єднання, захищене наскрізним шифруванням. Для створення прямого з’єднання між клієнтами використовуються технології STUN та hole punching.

STUN (Session Traversal Utilities for NAT) - мережевий протокол, що використовується для створення з’єднання в обхід NAT. Цей мережевий протокол дозволяє клієнту визначити свою зовнішню IP-адресу, спосіб трансляції адреси та порт у зовнішній мережі.

Hole punching (інколи - punch-through) - це методика, що використовується для створення прямого з’єднання між двома сторонами, де одна або обидві сторони знаходяться за фаєрволами або маршрутизаторами, що використовують NAT. Щоб “пробити дірку”, кожен клієнт приєднується до третьої сторони, яка не має на собі обмежень, на якій тимчасово зберігається інформація про зовнішні та внутрішні адреси та порти для кожного з клієнтів. Сервер передає цю інформацію від одного клієнту до іншого, після чого вони намагаються створити пряме з’єднання між собою. Оскільки для з’єднання використовуються чинні номери портів, фаєрволи та маршрутизатори приймають та пересилають вхідні пакети до іншої сторони.

Завдяки цьому набору модулів створюється взаємодія клієнт-сервер, frontend-backend та сервер-сервер. Усе це необхідно для створення працездатного програмного комплексу, який може працювати як у межах локальної мережі, так і через мережу інтернет.

Додатково слід згадати модуль ftplib, завдяки якому реалізується передавання файлів від одного клієнту до іншого. Цей модуль також входить до стандартної бібліотеки python і дозволяє обмінюватись файлами будь-якого типу.

Це необхідно для того, щоб користувачі мали змогу надсилати медіафайли, документи, архіви, та ще купу файлів різного виду, які можуть бути необхідними під час роботи над проектом у команді.

Як наслідок, ми маємо різноманітну та насичену мережеву систему, що забезпечує надійне функціонування менеджеру та дозволяє користувачам поширювати робочі файли і обговорювати робочий процес.

З точки зору криптографічної безпеки додатку, існує багато різноманітних шифрів, які дуже добре виконують свої задачі по забезпеченню надійності та секретності даних. Вибір конкретного шифру залежить від потреб та уподобань розробника. Якщо необхідна найвища криптографічна стійкість, а час шифрування не є пріоритетом, можна використовувати блочні шифри, як AES.

У випадку коли необхідна висока криптографічна стійкість, але питання безпеки також є важливим, можна обрати шифри асиметричної еліптичної кривої, такі як curve25519. Цей шифр дозволяє створювати надійний захист інформації, зберігаючи при цьому високу швидкість роботи програми.

Крива, що використовується у curve25519 є кривою Монтгомері, що визначена як

$$□^2 = □^3 + 486662□^2 + □ \quad (1.1)$$

де x, y - координати осі.

Ця крива базується на простих числах, що дозволяє уникнути атаки за алгоритмом Поліґа-Геллмана.

Алгоритм використовує стиснуту еліптичну точку (використовується тільки координата X), завдяки чому можливе ефективне використання Сходів Монтгомері для алгоритму еліптичної кривої Діффі-Геллмана, використовуючи тільки координати XZ.

До того ж, цей алгоритм був розроблений з урахуванням деяких можливих атак, завдяки чому він має імунітет до атак з визначенням часу та приймає будь-який рядок довжиною 32 байти як дійсний відкритий ключ та не потребує перевірки того, чи належить дана точка кривій або генерується базовою точкою.

Відсутність вразливості до атак з визначенням часу одразу позбавляє цей алгоритм однієї з найголовніших проблем, через яку деякі інші алгоритми не рекомендовано застосовувати при передачі інформації у мережі Інтернет, оскільки це може підставити під великий ризик її цілісність та таємність.

Підтримку цього алгоритму мовою програмування python реалізовано завдяки модулю, який має таку саму назву - curve25519.

Також, для простішого керування користувачами, повідомленнями та підключеннями до інших серверів, необхідно мати модуль, що дозволить працювати з базами даних. Python має велику кількість різних модулів, що працюють з різними типами баз даних. Серед підтримуваних баз даних є такі бази даних, як SQLite, MySQL та PostgreSQL.

Важливо зазначити що SQLite підтримується мовою Python за замовчуванням, не потребує окремого серверу для виконання операцій та дозволяє швидко і без проблем створювати, редагувати та видаляти таблиці, які можуть містити у собі безліч різної інформації.

В свою чергу, MySQL та PostgreSQL необхідно додатково завантажувати, а для виконання операцій з ними мати працюючий паралельно з основним додатком клієнт, що займає більше системних ресурсів.

Тому SQLite є очевидним вибором для задач цього дипломного проекту.

Розглянемо як за допомогою SQLite створювати та редагувати базу даних на прикладі простої бази даних користувачів.

Для цього ми визначаємо нову функцію, яку назвемо SQLUserAdd, що буде приймати на вхід наступні параметри: UUID користувача, логін користувача та хешований пароль.

```
def SQLUserAdd(uuid, login, pass):
```

Далі, всередині функції ми почнемо з перевірки того, чи існує взагалі база даних користувачів, для цього ми виконаємо наступний набір команд:

```
import os
path = 'data/users.db'
if !os.path.isfile(path):
     with open(path, 'w+') as f:
          pass
```

У випадку якщо файл не існує, ми його створимо та перейдемо далі, якщо файл існує то ми одразу почнемо виконувати наступну частину, а саме - внесення користувача до бази даних.

Виконуватиметься це за допомогою наступних команд:

```
conn = sqlite3.connect(path)
sql = ''' INSERT INTO users(uuid,login,password)
              VALUES(?,?,?) '''
cur = conn.cursor()
cur.execute(sql, [uuid, login, pass])
conn.commit()
```

Завдяки цьому набору команд ми створюємо новий запис у базу даних users.db, в якій зберігаються користувачі.

3.2 Створення з’єднань та користувацького інтерфейсу

Для створення з’єднань будуть використані перелічені раніше модулі, а саме - socket та libzt.

Щоб створити з’єднання за допомогою socket (тобто, з’єднання яке буде виконуватись в межах локальної мережі) необхідно виконати кілька кроків, а саме:

1. Визначити внутрішню IP-адресу серверу, яку необхідно буде зробити статичною, для того щоб при перезавантаженнях маршрутизатору не виникали проблеми, коли клієнт намагається підключитись до адреси, а на ній знаходиться інший клієнт замість серверу.
2. Визначити порт, через який буде виконуватись з’єднання, для прикладу оберемо порт 1550.
3. Написати код, за допомогою якого буде виконуватись підключення до серверу.
4. Виконати написаний код.

Виходячи з описаних задач, ми можемо почати створення коду підключення клієнту до серверу. У якості прикладу буде наведено роботу echo client + echo server.

Зі сторони серверу код буде виглядати наступним чином:

```
import socket
host = “192.168.0.105”
port = 1550
with socket.socket(socket.AF_INET, socket.SOCK_STREAM)
as s:
        s.bind((HOST, PORT))
        s.listen()
        conn, addr = s.accept()
        with conn:
            print(f"Connected by {addr}")
            while True:
                data = conn.recv(1024)
                if not data:
                    break
                conn.sendall(data)
```

Зі сторони клієнту код буде виглядати наступним чином:

```
import socket
host = “192.168.0.105”
port = 1550
with socket.socket(socket.AF_INET, socket.SOCK_STREAM)
as s:
          s.connect((host,port))
          s.sendall(b“Test message!”)
          data = s.recv(1024)
    print(f“Received {data!r}”)
```

Як результат виконання цього коду ми отримаємо відомості про те, хто підключився та що було надіслано зі сторони серверу та відповідь серверу зі сторони клієнту.

Далі треба встановити з'єднання через мережу з використанням zerotier. Усі дії, необхідні для цього, схожі з процесом з'єднання у попередньому прикладі, з невеликою різницею - необхідно мати зовнішню IP адресу, яку можна отримати на сайті zerotier.

Код клієнту буде виглядати наступним чином:

```
import libzt
remote_ip = ""
remote_port = 8080
client = libzt.socket(libzt.ZTS_AF_INET, libzt.ZTS_SOCK_STREAM, 0)
try:
   client.connect((remote_ip, remote_port))
   data = "Hello, world!"
   send_bytes = client.send(data)
   data = client.recv(1024)
   print("received:", data)
except Exception as ex:
   print(ex)
```

Код серверу матиме наступні відміни:

```
serv = libzt.socket(libzt.ZTS_AF_INET, libzt.ZTS_SOCK_STREAM, 0)
try:
   serv.bind("0.0.0.0", remote_port)
   serv.listen(5)
   while True:
         conn, addr = serv.accept()
         print("Accepted connection from:", addr)
         while True:
```

```
                    data = conn.recv(4096)
                    if data:
                        print(“recv:”, data)
                    elif not data:
                        break
                    conn.send(data)
                conn.close()
    except Exception as ex:
         print(ex)
```

Як можна побачити з цього прикладу коду, реальної різниці між процесом створення з’єднання між клієнтом і сервером у socket та libzt майже немає і більша частина цих різниць виходить з внутрішнього влаштування цих модулів.

Далі розглянемо приклад використання модулю eel. Цей модуль дозволяє запустити самостійне вікно браузеру, в якому буде відчинено вказану html сторінку, елементами якої можливо буде керувати завдяки прошарку у вигляді файлу eel.js, який буде пов’язувати між собою елементи html та виконуваний код python.

Для початку створимо просту html-сторінку, яка матиме кнопку, при натисканні на яку користувачу буде надаватись випадкове число.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta  http-equiv="X-UA-Compatible"  content="IE=edge"
/>
<meta   name="viewport"   content="width=device-width,
initial-scale=1.0" />
<title>Eel Example</title>
<style>
h1{
    color: green;
    text-align: center;
```

```
    }
    .random_number{
        margin: 50px;
        font-size: 150px;
        text-align: center;
    }
    button{
        display: block;
        margin: 0 auto;
    }
    </style>
    </head>
    <body>
    <h1>Random number:</h1>
    <div class="random_number"></div>
    <button>Get a Random number using Python</button>
    <script                          type="text/javascript"
src="../eel.js"></script>
    <script src="./script.js"></script>
    </body>
    </html>
```

З боку javascript файлу eel.js код матиме наступний вигляд:

```
    document.querySelector("button").onclick = function ()
{
        eel.random_python()(function(number){

document.querySelector(".random_number").innerHTML        =
number;
        })
    }
```

Цей код дозволить зв’язати нашу html сторінку з нашим кодом python, який ми розглянемо далі:

```
import eel
from random import randint

eel.init("web")

@eel.expose
def random_python():
     print("Random function running")
     return randint(1,100)

eel.start("index.html")
```

Для того, щоб eel знав до якої функції необхідно звернутись, використовується доданок `@eel.expose`, який, як виходить з назви, викриває цю функцію для eel.

При виконанні файлу python ми отримуємо наступне вікно (рисунок 3.1):

Рисунок 3.1 Початкова сторінка одразу після запуску програми

При натисканні на кнопку ми отримуємо результат у вигляді випадкового числа, як зображено на рисунку 3.2:

Рисунок 3.2 Результат виконання програми

Як бачимо, ми змогли створити зв’язок між нашою html сторінкою та нашим кодом python, завдяки чому ми можемо отримувати випадкові числа.

Засновуючись на цьому, було створено користувацький інтерфейс, до взаємодії з яким ми перейдемо далі.

3.3 Взаємодія користувача з системою

Коли користувач вперше запускає програму, йому необхідно авторизуватись для того, щоб почати користуватись месенджером. Для цього йому необхідно ввести юзернейм та пароль, які має надати адміністратор мережі. Вікно авторизації зображено на рисунку 3.3.

Рисунок 3.3 Вікно авторизації месенджера

Слід зазначити що при цьому, користувач має бути підключеним до корпоративної мережі інакше увійти в свій акаунт він не зможе.

Після успішної авторизації, користувач отримує доступ до самого месенджеру, у якому він вже зможе переписуватись з іншими користувачами.

Але спочатку вікно месенджеру буде пустим, як зображено на рисунку 3.4.

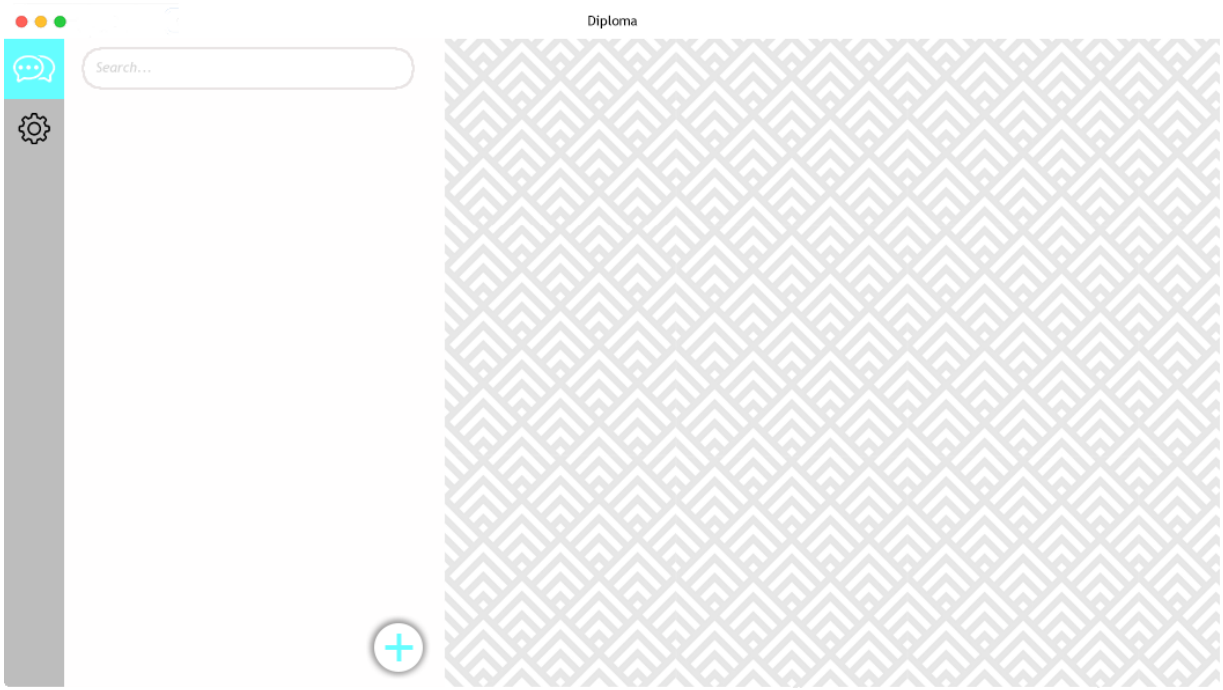


Рисунок 3.4 Початковий стан месенджеру

Коли користувач вперше запускає месенджер, в нього ще немає контактів або попередніх чатів, через що інтерфейс не має в собі нічого. Але, користувач може додавати нові контакти, якщо він здійснить пошук по юзернейму іншого користувача, що вже існує в мережі.

Для цього користувач повинен натиснути на іконку “+”, що знаходиться у нижній частині екрану, після чого з’явиться меню додавання контакту, як зображено на рисунку 3.5.

У цьому меню у користувача є наступні можливості:

1. Ввести юзернейм іншого користувача
2. Можливість попередньо побачити ім’я користувача та його зображення профілю
3. Якщо користувач існує, додати його за допомогою кнопки “Add”
4. Відмінити додавання користувача натисканням кнопки “Cancel”

Саме меню відображається у вигляді оверлею, з додаванням напівпрозорого чорного фону, який фокусує користувача саме на цьому меню.

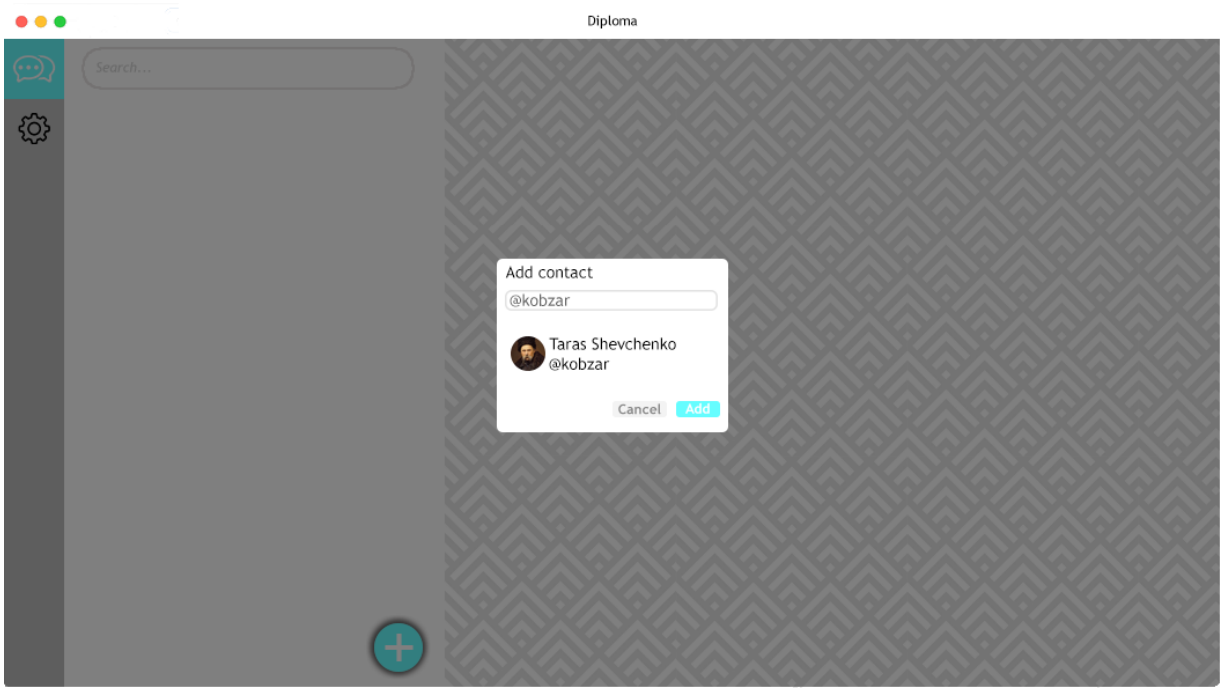


Рисунок 3.5 Меню додавання контактів

Після додавання контакту, він з’являється у бічному меню і у користувача з’являється можливість написати цьому контакту повідомлення.

Результат додавання контакту зображено на рисунку 3.6.

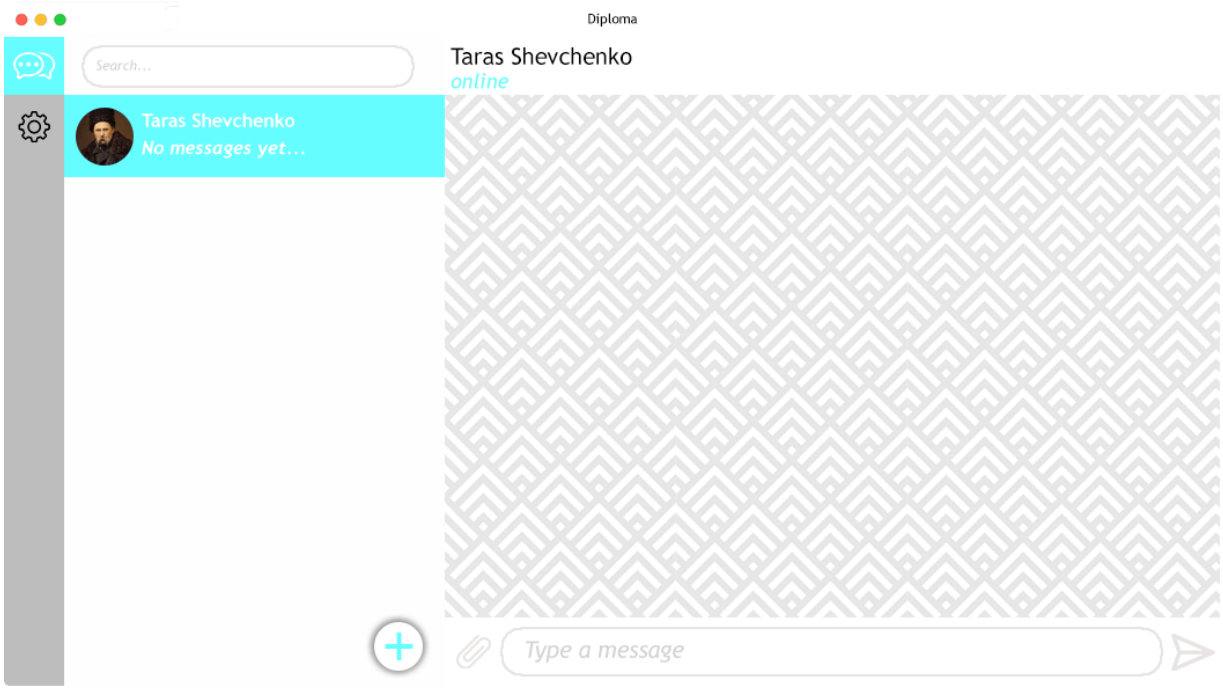


Рисунок 3.6 Результат додавання контакту

Після додавання контакту, користувач може одразу почати писати йому повідомлення, а також надсилати зображення та файли.

Крім цього, користувач може видалити історію листування, а також видалити або заблокувати певний контакт. Ця можливість передбачена для випадків, коли контакт замість роботи починає зневажливо відноситись до користувача, писати образи та інше. Альтернативно, користувач може поскаржитись своєму начальству та продемонструвати повідомлення, що містять образи, оскільки важливо зберігати здорову робочу атмосферу та не допускати приниження інших працівників. Меню взаємодії з контактом продемонстровано на рисунку 3.7.

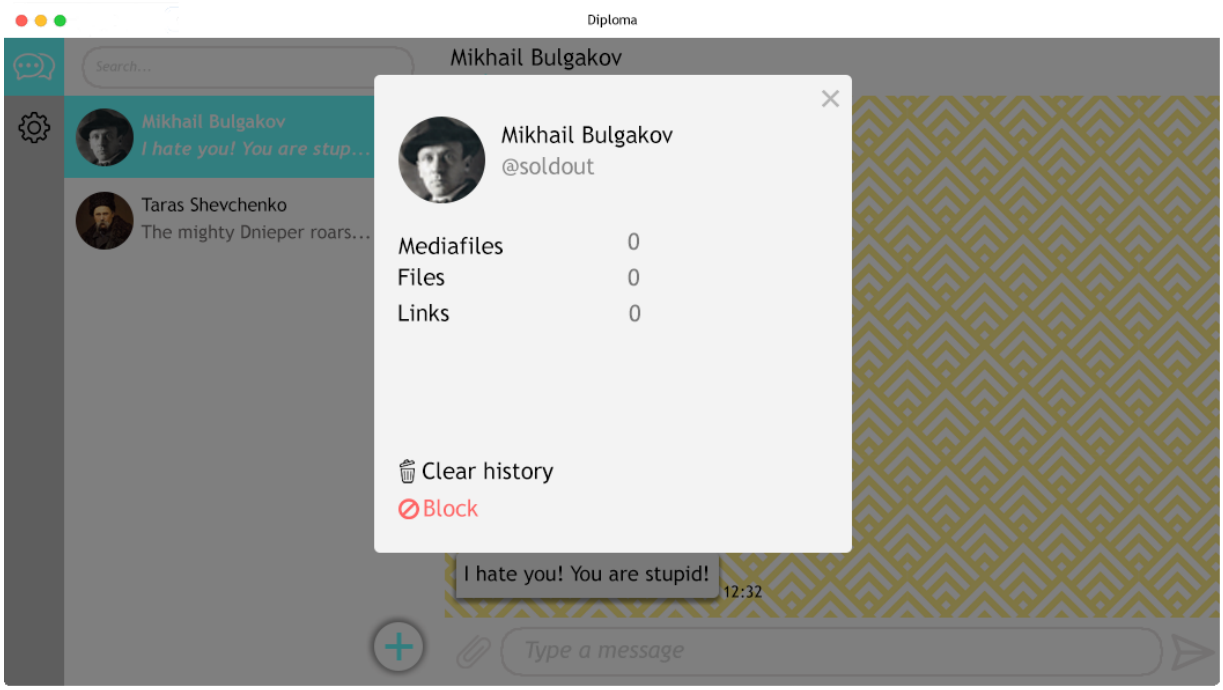
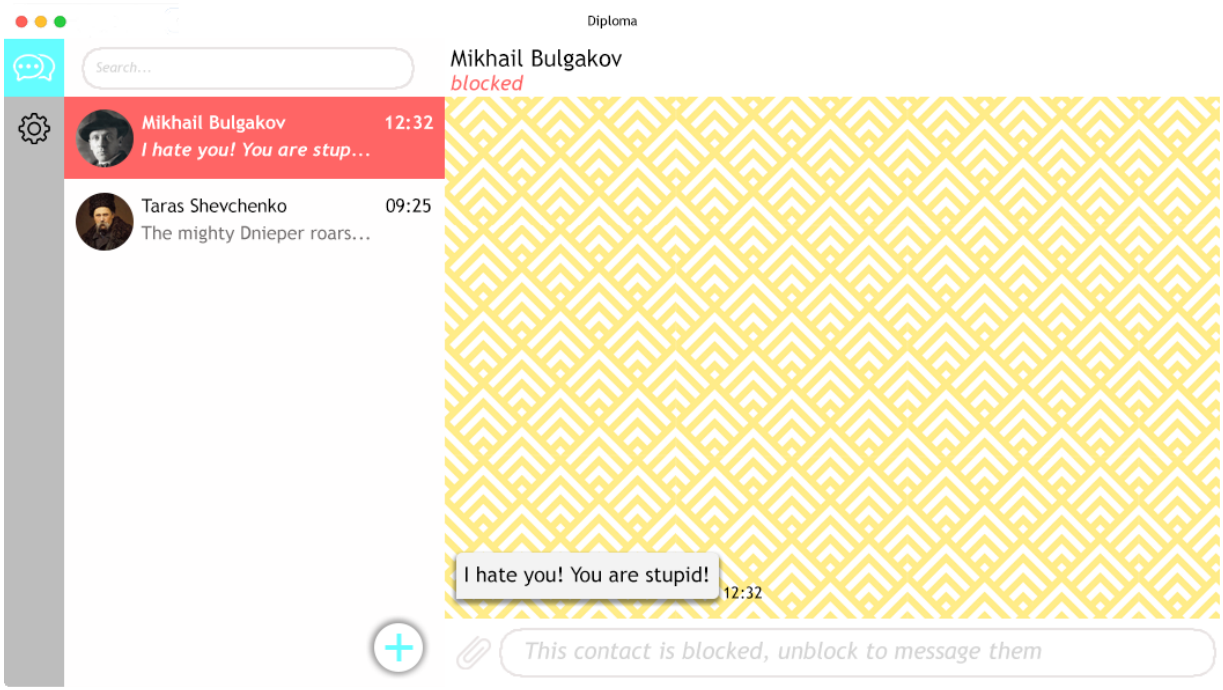


Рисунок 3.7 Меню взаємодії з контактом

Окрім можливості заблокувати неприємну особу, користувач також має можливість побачити всі фото/відео, файли та посилання, які є у листуванні з обраним контактом. Важливо зазначити, що при блокуванні користувач може не видаляти листування з контактом для збереження попередніх повідомлень, які можуть бути надані як докази порушення робочого етикету з боку контакту.

Вигляд листування з заблокованим контактом продемонстровано на зображенні 3.8.



3.7 Листування з заблокованим користувачем

Також користувач може скористатись налаштуваннями власного профілю, натиснувши на іконку шестерні, що можна побачити на зображенні 3.8.

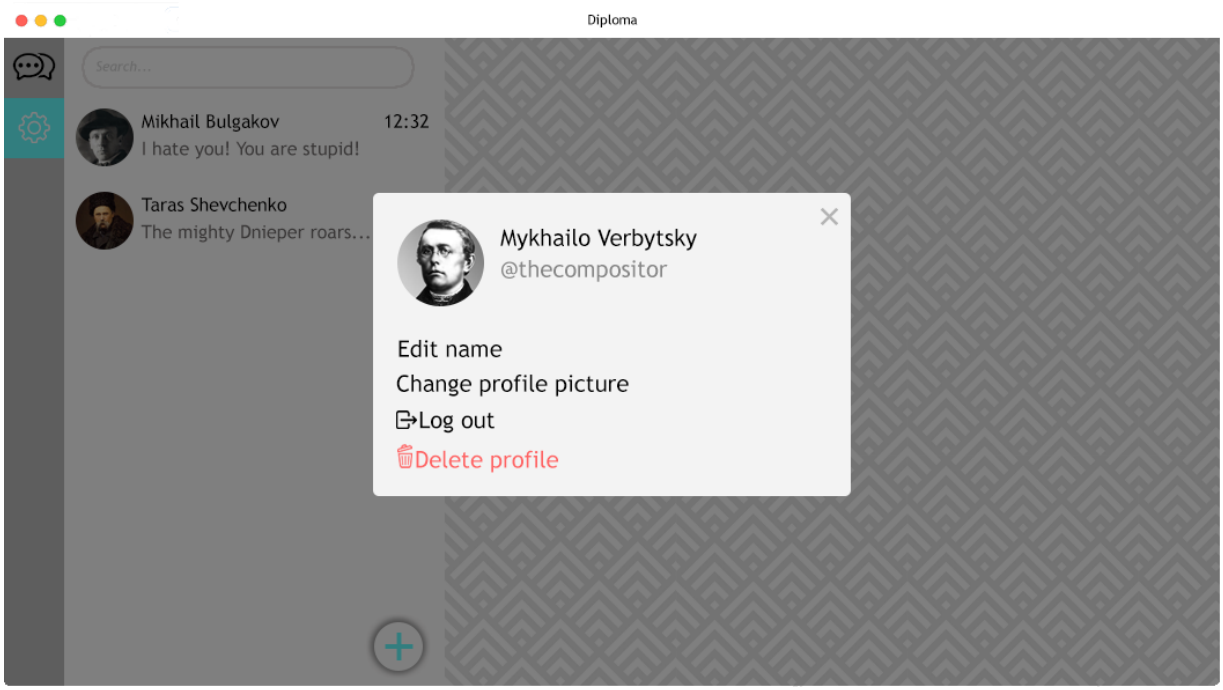


Рисунок 3.8 Налаштування профілю користувача

Користувач має можливість змінити ім’я та фото профілю, вийти з профілю або видалити його. Можливість зміни юзернейму недоступна користувачу, оскільки юзернейм задається адміністратором мережі під час реєстрації нового користувача.

З свого боку, адміністратор має можливість створювати нових користувачів, дані про яких будуть зберігатись у базі даних.

Форма реєстрації нового користувача зображена на рисунку 3.9.

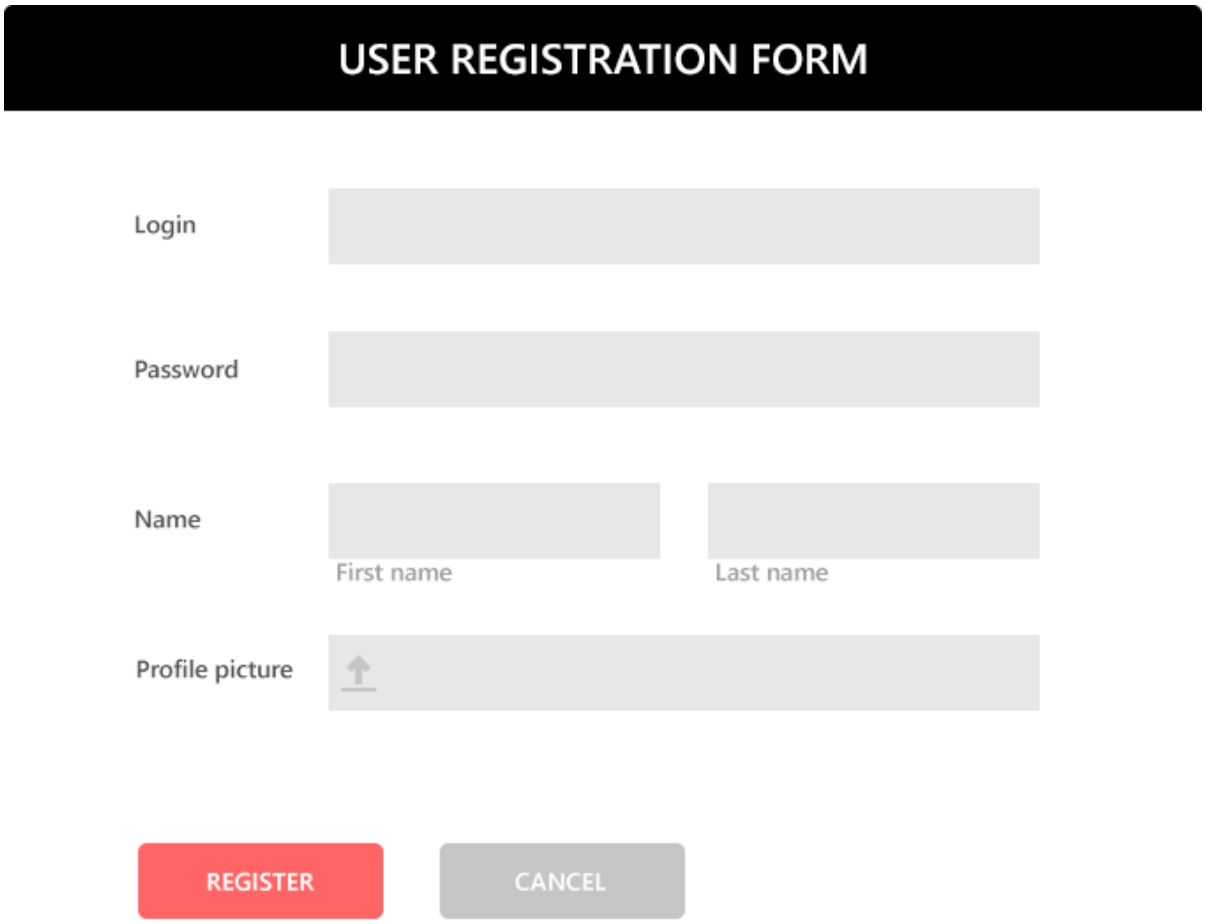


Рисунок 3.9 Форма реєстрації нового користувача

У цій формі адміністратор вказує наступне:

1. Логін
2. Пароль
3. Ім’я та Прізвище
4. Фото профілю

Можливість змінити Ім’я/Прізвище та Фото профілю надаються користувачеві на той випадок, якщо адміністратор зробить помилку при реєстрації користувача.

Висновок: у цьому розділі мною було розроблено алгоритм роботи системи месенджеру, розглянуто принцип роботи з’єднання між серверами, шифр, що використовується під час передачі даних, також мною було розглянуто процес створення користувацького інтерфейсу та продемонстровано як виглядає взаємодія користувача та програми.

ВИСНОВКИ

В результаті виконаної роботи з теми «Розробка мультисерверного корпоративного месенджера зі спеціальним захистом міжсерверного трафіку» було досліджено принцип роботи месенджерів в залежності від їх типу. Також, мною було розроблено месенджер, який на практиці може працювати як в межах одного, так і в межах декількох локальних мереж через мережу інтернет. Ця система може отримати широке розповсюдження та застосування як у корпоративних, так і у державних структурах, що дозволяє полегшити процес передачі інформації між окремими структурами. Контроль над такою системою виконується мережевими адміністраторами корпорації або держустанови.

ПЕРЕЛІК ПОСИЛАНЬ

1. LAN. Local Area Network. URL: https://en.wikipedia.org/wiki/Local_area_network
2. Pros and Cons of using LAN messenger for corporate. URL: https://potomya.net/benefits/advantages-and-disadvantages-of-lan-messenger/
3. Microsoft Teams. URL: https://www.microsoft.com/uk-ua/microsoft-teams/group-chat-software
4. MyChat. URL: https://nsoft-s.com/en/index.html
5. Slack. URL: https://slack.com/features
6. Microsoft Teams Pros and Cons. URL: https://www.syskit.com/blog/10-pros-and-cons-of-microsoft-teams/
7. Slack Pros and Cons. URL: https://texascreative.com/blog/pros-and-cons-slack
8. The Hole trick, Jurgen Schmidt, 2006. URL: http://www.h-online.com/security/features/How-Skype-Co-get-round-firewalls-747197.html
9. C++. URL: https://en.wikipedia.org/wiki/C++
10. Advantages and disadvantages of C++. URL: https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/
11. Python. https://en.wikipedia.org/wiki/Python
12. Advantages and disadvantages of Python. URL: https://data-flair.training/blogs/advantages-and-disadvantages-of-python/
13. JavaScript. URL: https://en.wikipedia.org/wiki/JavaScript
14. Advantages and disadvantages of JavaScript. URL: https://www.tutorialspoint.com/advantages-and-disadvantages-of-javascript
15. QT for Python. URL: https://doc.qt.io/qtforpython-5/contents.html
16. Eel in Python. URL: https://javatpoint.com/eel-in-python
17. HTML & CSS. URL: https://www.w3.org/standards/webdesign/htmlcss
18. Advantages and disadvantages of HTML and CSS. URL: https://www.aplustopper.com/advantages-and-disadvantages-of-html/
19. ZeroTier. Protocol design whitepaper. URL: https://docs.zerotier.com/zerotier/manual

20. Curve25519. URL: https://en.wikipedia.org/wiki/Curve25519